The classes P and NP

Reading

Section 7.2

Practice problems (page 294):

The class P

Computational problems are considered “tractable”, if their running time in polynomial. While a polynomial can grow with n, it does so in much more reasonable ways than an exponential.

> The class P consists of all languages that are decidable in polynomial time by a deterministic single-tape Turing Machine.

A number of well-known problems belong to the class P, and looking back at algorithms you have learned in your other classes you can find more examples.

We will now consider several popular members of the class P

The Path problem

The path problem is represented by the language:

$$\text{PATH} = \{\langle G, s, t\rangle \mid G \text{ is a directed graph and has a directed path from } s \text{ to } t\}$$

$$\text{PATH} \in P$$

One relevant question is how we represent the graph G. There are various ways, and they all involve space polynomial in the number of nodes n. Since the class P is effectively invariant under such transformations, we can consider n to be the “size” of our problem.

A “naive” approach to solving this problem would attempt to consider all possible “paths”, which are m^m if we denote by m the number of edges. This would not be polynomial (m is essentially $O(n^2)$).

But of course there are more efficient ways, essentially involving marking of the vertices:

1. Start by marking the vertex s.

2. Repeat until nothing new is marked:
   - Go through the edge list.
   - If the source is marked and the target is not, mark the target.
3. See if t is marked.

Clearly the time-intensive portion is the second part. It will have to run once for each vertex (because each time it must mark a new vertex or we are done), and it takes time $O(m)$ to run through the edge list. So it's total running time is $O(n^3)$, polynomial.

Relatively prime numbers

Another popular polynomial-time problem is the determination of whether two numbers are relatively prime or not. An important consideration here is the size of the input.

> A number N is stored in base 2 using $n = O(\log N)$ space, by simply using the base-2 representation of N.

Now consider the problem:

$$\text{RELPRIME} = \{\langle x, y\rangle \mid x, y \text{ are relatively prime }\}$$

$$\text{RELPRIME} \in P$$

The size of the input is here $O(\log N)$ where N is the largest of the two numbers. This is important to keep in mind. For instance a naive approach would be to go through each number d up to x, y and divide into them to see if it is a common factor. But this would take too long: There are in general $O(N)$ such numbers, and $N = 2^{O(n)}$ is exponential in the size of the input. Essentially, it would take too long.

Instead we will perform the well-known Euclidean division algorithm:

> E = On input $\langle x, y\rangle$:
>
> A. Repeat until $y = 0$: 1. Compute $x = x \bmod y$ 2. Swap x and y B. The resulting x is the greatest common divisor of x, y. If it is 1 then we accept, otherwise we reject.

The key intuition here is that each repetition is effectively cutting the size of the inputs x, y by at least a half every second time through the loop. So the number of A steps needed is $O(\log N)$. Each of those steps is also polynomial in $\log N$, the length of the representations of x, y.

TODO: Work in progress